VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 **Martin Jeremiáš**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student:	**Martin Jeremiáš**
Studijní program:	B2647 Informační a komunikační technologie
Studijní obor:	2612R025 Informatika a výpočetní technika
Téma:	Absolvování individuální odborné praxe Individual Professional Practice in the Company
Jazyk vypracování:	čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Tieto Czech s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Seidl, Ph.D.**

Konzultant bakalářské práce: Ing. Elzbieta Stacherska

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * 1D16 * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 29, odst. 9 Studujního a zkušebního řádu pro studium v bakalářských programech VŠB-TUO Ostrava.

V Ostravě dne 21.4. 2016

.....
podpis

Tieto Czech s.r.o.
28. října 3346/91
702 00 Ostrava - Moravská Ostrava
IČO 64608051 DIČ CZ64608051

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 21.4. 2016

.....
podpis

Poděkování

Rád bych poděkoval svému vedoucímu panu Ing. Davidu Seidlovi, Ph.D. za odbornou pomoc při vytváření této bakalářské práce.

Dále bych chtěl poděkovat svým kolegům, se kterými jsem měl možnost spolupracovat a kteří mě podporovali v absolvování odborné praxe ve firmě Tieto Czech s.r.o.

Abstrakt

Tato bakalářská práce popisuje individuální odbornou praxi, kterou jsem absolvoval ve firmě Tieto Czech s.r.o. Cílem této praxe bylo využití teoretických a praktických znalostí ze studia v praxi a zlepšení teoretických a praktických znalostí a dovedností. Popisuji zde zaměření firmy a mou funkci, kterou jsem vykonával v průběhu mé praxe. Dále jsou popsány použité technologie, programovací jazyky a poté jsou popsány jednotlivé úkoly, jež jsem vypracoval a jejich řešení. Další část je věnována mým dosaženým výsledkům a chybějícím dovednostem. Na závěr zhodnocuji odbornou praxi a její výhody při jejím absolvování.

Klíčová slova

Odborná praxe, praxe ve firmě, Tieto Czech s.r.o., JIRA, BSA, BMC, PowerShell, NSH, automatizace

Abstract

This bachelor work describes individual professional practice, which I attended in company Tieto Czech s.r.o. The purpose of this practice was to use theoretical and practical knowledge from the study in practice and improve theoretical and practical knowledge and skills. Here I describe the focus of the company and my operation I performed during my internship. The following describes the technology used programming languages and then describes the different tasks that I developed and their solutions. Another section is devoted to the results and my missing skills. At the end I evaluate professional practice and its benefits in its completion.

Keywords

Professional practise, practise in company, Tieto Czech s.r.o., JIRA, BSA, BMC, PowerShell, NSH, automation

Obsah

Seznam zkratek

ABX – ABX Automation Framework

BAO – BMC Atrium Orchestrator

BSA – BMC Server Automation

ISE – Integrované skriptovací prostředí (Integrated scripting environment)

IT – Informační technologie

KMs – Knowledge modules

NSH – Network Shell

PA – BMC Patrol Agent

RDP – Remote Desktop Protocol

UC – Use Case

XML – Extensible Markup Language

Seznam obrázků

1 Úvod

Bakalářskou práci jsem vykonával formou individuální odborné praxe ve firmě Tieto Czech s.r.o. Tato možnost se mi naskytla díky tomu, že už jsem v Tieto Czech s.r.o. pracoval, ale na jiné pozici, která neměla s programováním nic společného. Po krátkém přemýšlení jsem dal přednost této možnosti před klasickou bakalářskou prací, protože jsem chtěl získat jak teoretické, tak hlavně praktické dovednosti z reálného života.

V bakalářské práci popisuji odborné zaměření firmy Tieto Czech s.r.o. a mé pracovní zařazení společně s pracovní náplní. Dále se zabývám popisem technologií, kterou jsem použil a prohloubil si jejich znalosti. Potom popisuji jednotlivé úkoly, které jsem měl vyřešit, a jejich řešení. V neposlední řadě se věnuji znalostem a dovednostem, které jsem získal v rámci studia a v průběhu odborné praxe, a které jsem postrádal. V závěrečné části práce popisuji mé dosažené výsledky a zhodnocení odborné praxe.

2 Popis odborného zaměření firmy a mého pracovního zařazení

2.1 O společnosti Tieto Czech s.r.o.

Tato společnost byla založena spojením finské společnosti Tieto Czech s.r.o. a švédské společnosti Enator dne 7. 7. 1999, s hlavním sídlem v Helsinkách ve Finsku. V roce 2001 vstoupila do české republiky. Roku 2004 otevřela softwarové centrum v Ostravě. Má více než 15 000 zaměstnanců a působí ve více než 20 zemích, převážně v Evropě.

Tieto Czech s.r.o. je společnost poskytující služby hlavně v IT oblasti, výzkumu, vývoje a poradenství. Mezi hlavní trhy patří severní Evropa, Německo a Rusko. Na těchto trzích se zaměřuje v poskytování služeb velkým a středně velkým organizacím. Celosvětově spolupracuje se svými zákazníky v oblasti telekomunikačních a digitálních služeb, v lesním, ropném a plynárenském průmyslu. Svým zákazníkům poskytuje služby v oblastech bankovnictví, financí a pojišťování, zdravotní a sociální péče, logistiky, energetiky a další.

Zaměřuje se na oblasti, ve kterých má ty nejdůkladnější znalosti podnikání a potřeb jejich zákazníků. Díky své odborné činnosti se řadí mezi největší poskytovatele IT služeb a je přední světovou společností ve vybraných odvětvích.

2.2 Pracovní zařazení studenta ve firmě

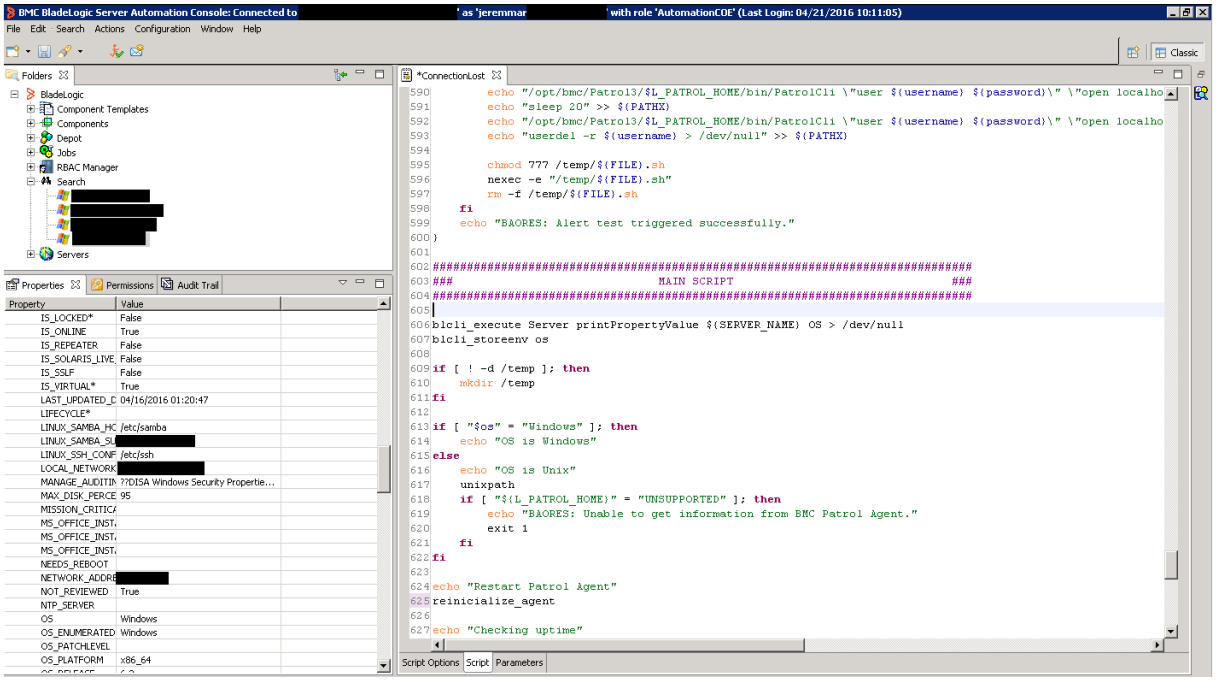
Ve firmě jsem byl přiřazen do oddělení Automation Server Management, které spolupracovalo s ostatními odděleními. V našem oddělení byly převážně interní projekty. Zautomatizovávali jsme zdlouhavé, opakující se a časově náročné práce druhým oddělením, a díky tomu se ušetřil jejich čas a mohli se věnovat jiným problémům.

3 Použité technologie

3.1 BMC technologie

BMC BladeLogic Server Automation

BSA je řešením pro automatizované řízení, kontrolu a provádění konfiguračních změn na serveru. Poskytuje správci přístup, aby mohl spravovat data serverů, administratorské zabezpečení, konfigurování, aktualizace, udržování fyzických, virtuálních a cloudových serverů, vyhledávat servery, upravovat jejich informace a dalších mnoho věcí, které se týkají správy a nastavení. Také umožňuje vytvářet a editovat NSH skripty, ze kterých se vytváří job[1], který se spouští na serveru nebo více serverech najednou. Ukázka práce v prostředí BSA na obrázku 1. Použité zdroje [1], [3].



Obrázek 1 - BMC BSA Console

ABX Automation Framework

Tento framework je vyvinut s použitím BMC Atrium Orchestrator podporovaný webovou aplikací sloužící k uložení informací o automatizaci a podrobnosti o výsledcích. Hlavním cílem tohoto frameworku je zajistit flexibilní, dynamickou a daty řízenou automatizaci s opakovaně použitelnými komponenty, které snižují vývojové a testovací potřeby s krátkou dobou zařazení do ostrého nasazení. Díky tomuto frameworku se dají nasazovat nové standartní automatizační Use Cases[2] během pár minut a to tím, že se použije již vytvořený kód v BAO, který se dá parametrizovat.

[1] job – vytvořený z NSH skriptu, který se spouští pomocí BAO na zvoleném serveru
[2] Use Cases – případ užití je seznam kroků, který se definuje interakci s tzv. rolí a systémem

ABX vytváří XML[1] se všemi instrukcemi, které se mají na straně BAO provést. Za pomocí definice ABX se dá reagovat např. na tikety v ServiceNow[2] a spouštět za pomocí BAO integraci různé joby skrz BSA na určitém serveru. Tím, že se používají standardně vytvořené BAO workflows[3], nedochází při nasazování nového UC k chybám samotného workflow, který je již otestovaný. Ukázka prostředí ABX na obrázku 2. Použité zdroje [2], [3], [8].

Výhody: 1) Rychlé zpracování k použití (quick time-to-production)

- Při použití ABX a již dostupných funkcí, rychlé zpracování k použití trvá asi 10 minut týkající se orchestrace[4]. Jakékoliv funkce, které se přidají do ABX jsou okamžitě dostupné ke znovu použití pro další workflows.

2) Snížené testovací potřeby (low testing needs)

- Opětovné použití komponent znamená, že není potřeba opakovaného testování daného workflow, každá část workflow je již otestováná a tudíž nemusí být znovu otestováná pro každý UC.

3) Menší množství servisních přestávek (lower amount of service breaks)

- každá změna modulu uvnitř BAO je příčinou přestávky. Spuštěné procesy jsou jednoduše ukončeny BAOem a znovu zapnuty. S ABX není potřeba žádné přestávky, změny jsou prováděny pouze uvnitř webové aplikace.

4) Plná viditelnost do orchestrace (full visibility into orchestration)

- Úplný přehled o tom co je právě spuštěno, co bylo spuštěné a jaký byl výsledek.

5) Kompletní odchycení chyb (complete error handling)

- Každá chyba je zaznamenáná v ABX webové aplikaci s možností analyzovat vstupní pracovní proces k nalezení slabých míst a tím zlepšit úspěšnost. Přes ABX můžeme indentifikovat různé závažné problémy v BAO a BSA, které nejsou vidět před spuštěním a ovlivňují ostré nasazení.

6) Schopnost zastavit orchestraci (ability to stop orchestration)

- V případě. že dojde k servisní přestávce pro tiketovací systém, BSA atd. je velmi snadné dočasně zastavit všechny pracovní procesy běžící přes ABX, aby se zabránilo špatnému provedení pracovních procesů. BAO nemá takovou schopnost.

---

[1] XML – obecný rozšiřitelný značkovací jazyk, který umožňuje vytváření konkrétních tzv. aplikací pro různé účely a různé typy dat

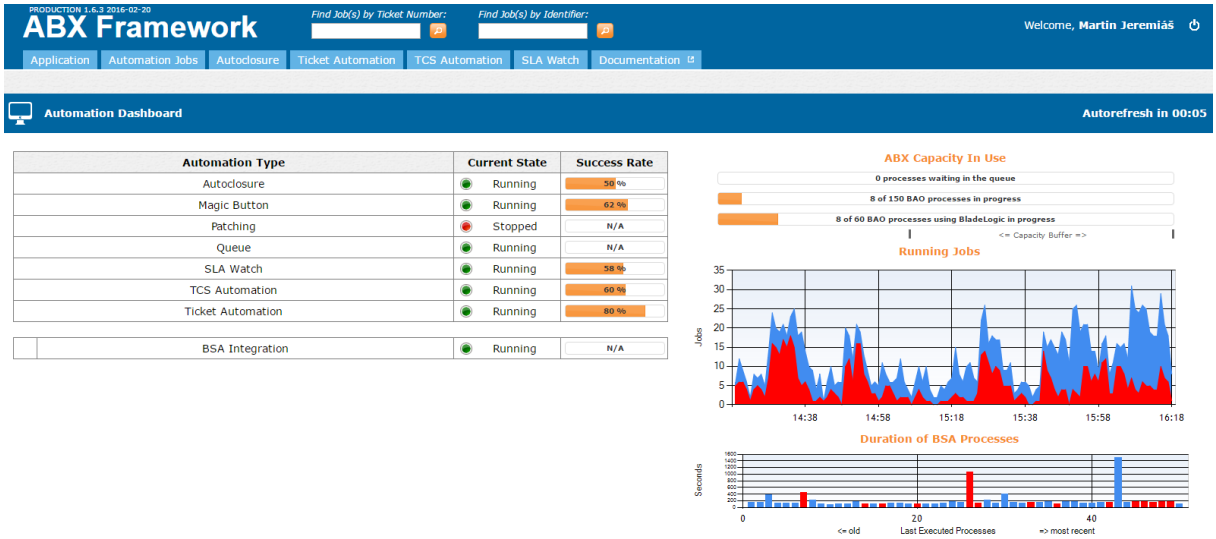
[2] ServiceNow – tiketovací nástroj s webovým rozhraním a databází s informacemi, které se používají v tiketech (jméno serveru, zákazník atd.)

[3] workflow – pracovní postup

[4] orchestrace – popisuje automatickou koordinaci a řízení komplexních počítačových systémů, middleware a služeb. Zabezpečuje koordinaci procesů a výměnu informací pomocí webových služeb

7) Reálné monitorování (real-time monitoring)

- Přes ABX webovou aplikací je možné zachytit multifunkčnost BAO, BSA atd. s téměř nulovým zpožděním a mnohem rychlejší než monitorování. Někdy monitorování zobrazuje všechno v pořádku, ikdyž automatizace nefunguje.



Obrázek 2 - ABX framework webová prostředí

BMC Atrium Orchestrator

Je IT proces automatizovaného řešení, které umožňuje IT pracovníkům vytvářet workflows v rámci aplikací, platforem a nástrojů k orchestraci kritických aktivit pro vyhovění, zabezpečení a konkurenční výhodu. Umožňuje nahlídnout na obraz a automatizační úkoly pomocí workflows, které se klenou více aplikacemi, systémy a infrastrukturou s méně zdrojů a vyšší kvalitou, předvídatelných výsledků. Snižuje chyby nebo výpadky způsobené manuálním úsilím nebo ručně psanými skripty, které se pokusí svázat neintegrované systémy dohromady. Tím zaručuje větší stabilitu a menší dopad na změny pro celý systém. Umožňuje orchestraci mezi ServiceNow, BSA, ABX a další. Použité zdroje [2], [3], [8].

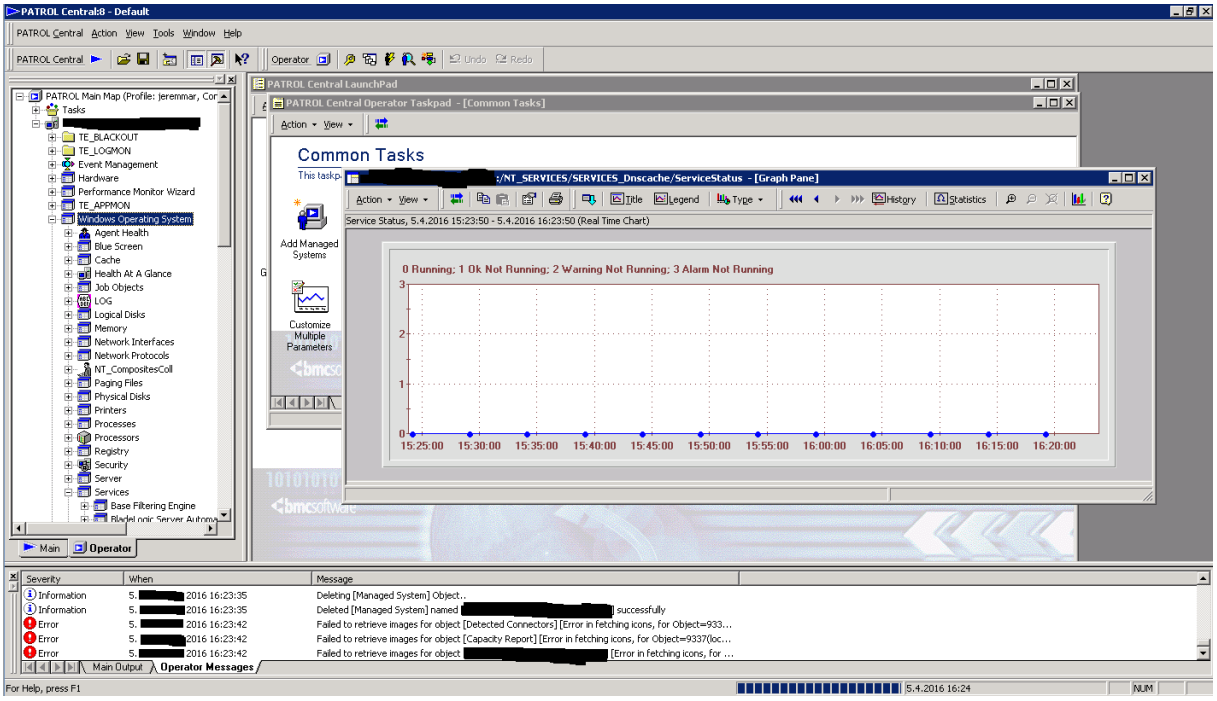
Hlavní rysy:

- zvyšuje produktivitu IT technologií a podnikání
- vizualita - představuje end-to-end podnikační proces automatizace
- rychlost - orchestrace dělá práci a uvolňuje zdroje pro důležitější úkoly
- flexibilita - umožňuje změny ve svém prostředí, aniž by došlo k přepisování workflows
- automatizace - eliminuje práci a snižuje potřeby zásahu člověka

BMC Patrol Agent

Patrol Agent je jedna z komponent Patrol aplikační řídící sady od BMC, která se skládá z mnoha komponent pro sledování různých softwarových aspektů na serveru. Nejvíce používanější je Patrol Agent, který je monitorovací komponenta. Využívá „knowledge modules“, pomocí kterých se určuje co

monitorovat, jak detekovat, shromážďovat a oznamovat různé aktivity na serveru. Klíčem k přístupu Patrol je relativní samostatnost Patrol Agenta, který při načtění s instrukcemi z „knowledge modules“ může startovat, monitorovat, zaznamenávat a spravovat server bez připojení konsole. Ukázka prostředí Patrol Agenta na obrázku 3. Použitý zdroj [8].



Obrázek 3 - BMC Patrol Agent, ukázka monitorování služby DNS Clienta

3.2 JIRA

Tento software umožňuje zlepšení efektivity práce podporou a usnadněním procesu řízení a sledování projektů, úkolů nebo požadavků. Nabízí flexibilní a uživatelské nástroje pro řízení a sledování pracovníků při výkonu plnění úkolů. Je orientován na podporu dosažení očekávaného výkonu na projektu. Do tohoto systému se mohou zapisovat jednotlivé chyby, nové projekty, požadavky, vylepšení a další. Ukázka prostředí JIRA na obrázku 4. Použitý zdroj [10].

Hlavní výhody: - podpora projektového řízení (interní a externí řízení požadavků a úkolů)

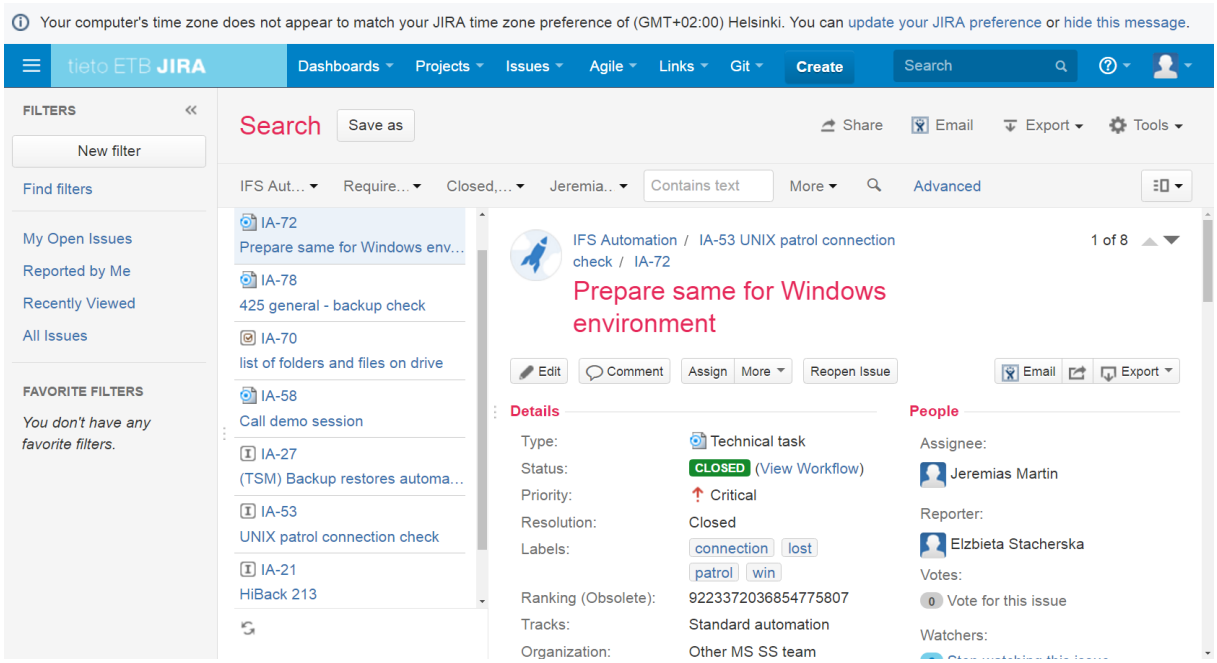
- workflow management
- neustálé dostupné informace pro tým přes webové rozhraní
- sledování a vyhodnocování kapacit
- průkazná historie projektové komunikace
- podpora pro klientský servis a helpdesk
- sdílení komunikace, informací a dokumentů v týmu
- reporty, statistiky, historie evidence

- sledování stavu projektů a řešení požadavků zákazníkem

- úkoly podle priorit a termínů dokončení

- fulltextové vyhledávání[1], rozšířené filtrovací nástroje

- projektové statistiky



Obrázek 4 - Webové prostředí JIRA

3.3 Programovací jazyk PowerShell

Je to rozlišitelný Shell[2] se skriptovacím jazykem od společnosti Microsoft. Produkt je založen na platformě .NET Framework[3] a z toho vyplývá i jeho odlišnost od ostatních shellů. Místo textové struktury, jak je tomu u UNIX shellu, obsahuje PowerShell strukturu objektovou. PowerShell poskytuje všechny možnosti platformy, na které je postaven, tudíž vše, co je obsaženo v Microsoft .NET Framework, je dostupné i z PowerShellu. Díky této provázanosti poskytuje PowerShell velké množství funkcí pro správu pomocí tzv. cmdlets, což jsou specializované třídy .NET implementující určitou operací.

PowerShell obsahuje dynamický skriptovací jazyk, který podporuje složitější imperativní operace pomocí cmdletů. Skriptovací jazyk podporuje proměnné, které nemají typovou kontrolu, dále jsou v jazyce podporovány funkce, větvení, cykly, strukturované odchycení výjimky a lambda výrazy. Použitý zdroj [9].

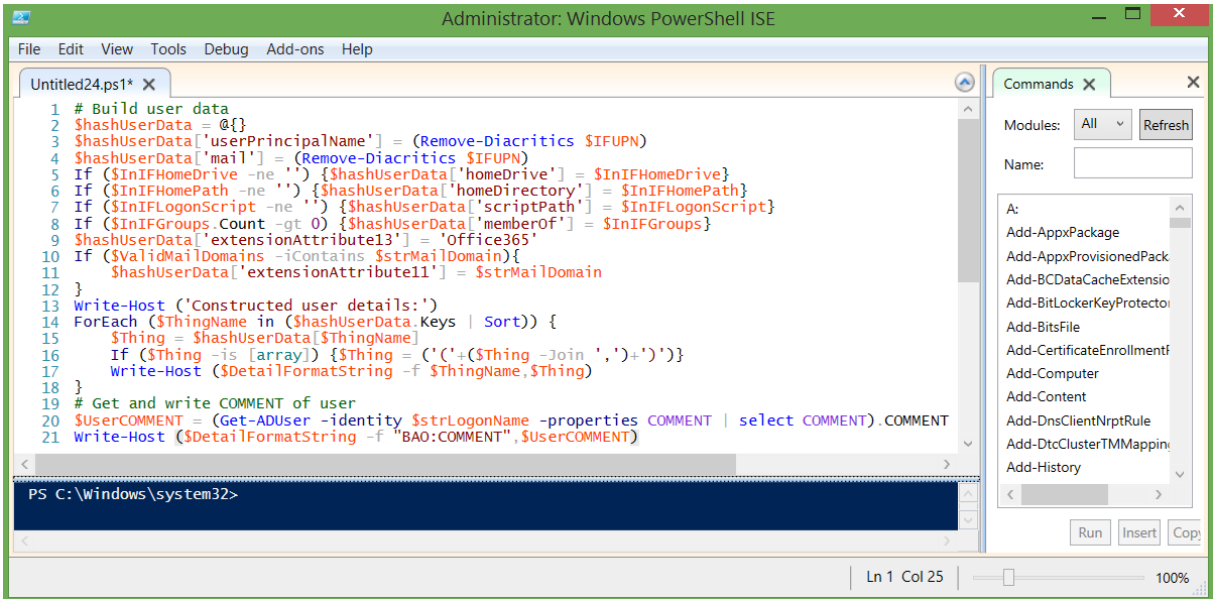
[1] fulltextové vyhledávání – je speciální způsob vyhledávání informací v databázích nebo v textových souborech, které jsou obvykle předem připraveny tj. Indexovány, aby bylo možno nalézt libovolné slovo nebo řetězec znaků v nejkratším možném čase

[2] Shell – je vedomě kontextové doplňování kódu funkce v programovacím prostředí, které urychluje proces kódování aplikací

[3] .NET Framework – prostředí potřebné pro běh aplikací a nabízející jak spouštěcí rozhraní, tak potřebné knihovny

3.4 Windows PowerShell Integrované Skriptovací Prostředí

Windows PowerShell ISE (dále jen PS ISE) je zdarma volně dostupná hostitelská aplikace s uživatelským rozhraním pro prostředí Windows PowerShell. V aplikaci PS ISE se můžou psát a spouštět příkazy, testovat a ladit skripty způsoby, které nejsou k dispozici v aplikaci Windows PowerShell konzoli. PS ISE přidává syntaxe zbarvení, multi editaci, selektivní spouštění, podporu pro jazyk z prava do leva, panel karet, IntelliSense[1], vizuální ladění a kontextovou nápovědu. PS ISE umožňuje spouštět příkazy v okně konzole, ale také podporuje panely, které se můžou použít současně s prohlížením zdrojového kódu svých skriptů a dalších nástrojů, které lze připojit do PS ISE. Také se může otevřít více oken skriptu najednou, což je zvláštně užitečné při ladění skriptů, který používá funkce definované v jiných skriptech nebo modulech. Ukázka aplikace Windows PS ISE na obrázku 5. Použitý zdroj [9].



Obrázek 5 - Windows PowerShell ISE, ukázka PowerShellového skriptu

3.5 Programovací jazyk Network Shell

Programovací jazyk NSH je rozhraní příkazové řádky založeno na rozšiřitelném Z-Shellu (ZSH), který může být použit jako interaktivní přihlašovací shell anebo jako mocný příkazový interpret pro psaní shellových skriptů. ZSH si můžeme představit jako rozšířený Bourne shell[2] s velkou řadou vylepšení, včetně některých z nejužitečnějších funkcí Bashe, Korn shellu a TENEX C shellu.

BSA používá NSH ke komunikaci se vzdálenými servery různých operačních systémů. Poskytuje podporu pro mnoho Windows a Unix skriptů a knihovnu s více než 130 síťovými příkazy, které jsou univerzálními s RSCD agenty[3] v systému AIX, HP-UX, Linux, Solaris a Windows systémy.

[1] IntelliSense – je vedomě kontextové doplňování kódu funkce v programovacím prostředí, které urychluje proces kódování aplikací
[2] Bourne shell – byl standardním unixovým shellem pro systém Unix, který byl uvolněn v roce 1977 pro školy a univerzity
[3] RSCD agent – aplikace, která umožňuje přístup pro BSA ze vzdáleného počítače

Můžou se použít příkazy NSH pro psaní nových skriptů nebo modifikovat již existující skripty. Pomocí těchto příkazů se můžou spravovat sítě počítačů, provádět systémové funkce pro správu na více vzdálených počítačů z jednoho počítače a přistupovat k souborům na lokálních a vzdálených počítačů přímo z příkazové řádky. Použité zdroje [3], [5].

Hlavní výhody: - programovatelné doplňování/dokončování parametrů a voleb pro nejpoužívanější příkazy, řešení tzv. „out-of-box“ s podporou pro stovky příkazů

- sdílení historie příkazů pro všechny spuštěné shelly
- vylepšené zpracování proměnných a datových polí
- editování víceřádkových příkazů v jednom bufferu
- kontrola správnosti zápisu příkazů
- různé režimy kompatibility, například NSH se může chovat jako Bourne shell, když je spuštěn jako „/bin/sh“
- plná přizpůsobitelnost

```
echo "Getting information from server '${SERVER_NAME1}' for execute script.
# Set powershell script location
blcli_execute DepotFile getLocationByGroupAndName "${PS_SCRIPTS_LOCATION}"
if [ $? -ne 0 ]; then
    echo "ERROR: Can not get PowerShell file location, check name and BSA c
    blcli_disconnect
    exit 1
fi
blcli_storeenv LOC_PSSCRIPT1

# Get staging dir value
blcli_execute Server printPropertyValue "${SERVER_NAME1}" STAGING_DIR >/dev
if [ $? -ne 0 ]; then
    echo "ERROR: Can not get STAGING_DIR value."
    blcli_disconnect
    exit 1
fi
blcli_storeenv stagedir
if  [ -z "${stagedir}" ]; then
    echo "ERROR: N/A - Unable to check. (BSA Property 'STAGING_DIR' is empt
    blcli_disconnect
    exit 1
fi
```

Obrázek 6 - Ukázka NSH skriptu

4 Úkoly v průběhu praxe

Po nástupu a seznámení s oddělením jsem dostal vlastní notebook, který jsem si měl zprovoznit a nastavit si programy, které budu používat. Nastavování netrvalo dlouho, protože se jednalo pouze jeden program a to RDP[1], pomocí kterého jsem se připojoval na administrátorský server, který měl nainstalované programy BSA a PA, se kterými jsem pracoval. Jakmile bylo vše hotové, tak jsem měl za úkol projít kurzem, který se věnoval programování v NSH. Po ukončení kurzu mi byly zadány úkoly. Všechny úkoly se zapisovaly do JIRA, aby se vědělo, co dělám a v jakém stavu je daný úkol. V případě problému se zadaným úkolem jsem vždy věděl, na koho se můžu obrátit a na koho směřovat případné dotazy. Při složitějších problémech mi bylo řečeno, jak mám postupovat při řešení a co mám v daném zdrojovém kódu upravit anebo jsem dostal část zdrojového kódu, který jsem rozšiřoval. Součástí úkolů bylo také vypracování dokumentace.

4.1 Seznam souborů a složek na disku

4.1.1 Zadání

Do začátku jsem dostal lehčí úkol, jehož zadáním bylo vypsání všech souborů a složek na požadovaném disku a složce na zadaném serveru, který byl v tiketu. Pomocí ABX se bude spouštět skript a následně zapisovat výsledek výpisu do poznámek k tiketu. V tomto úkolu jsem se měl také seznámit a naučit se pracovat s programem BSA.

4.1.2 Řešení

Nejtěžší bylo naučit se ovládat program BSA. Z počátku jsem si nevěděl rady, ale po zasvěcení kolegou do problematiky jsem přesně věděl co dělat. Poté jsem se pustil do skriptování samotného kódu. Bylo nutné, abych se naučil a prohloubil si znalosti týkající se technologie NSH, dále jsem použil příkazy z příkazové řádky Windows. Po dokončení kódu a odzkoušení skriptu v BSA jsem narazil na problém s výpisem správných souborů a složek. Po chvíli zjišťování jsem zjistil, že problém byl ve vstupním zadávaném parametru, který jsem následně opravil přeformátováním řetězce. Po skončení skriptu se vše vypisuje do poznámek k tiketu.

DRIVE=${DRIVE//\>/\\} - záměna znaku „>“ za „\“

DRIVE=${DRIVE//\</\ } - záměna znaku „<“ za mezeru

DRIVE=${DRIVE//>/\\} - záměna znaku „>” za “\”

[1] RDP – Remote Desktop Protocol je proprietární síťový protokol, který umožňuje uživateli ovládat vzdálený počítač prostřednictvím připojení k jeho desktopovému prostředí

4.2 Ztracené spojení

4.2.1 Zadání

Dalším můj úkol byl již složitější. Na základně tiketu, který obsahoval informace o chybě se špatnou dostupností serveru, jsem měl vytvořit skript. I když bude server v pořádku, přesto může přijít takový probém tiket a to z toho důvodu, že PA přestal pracovat a tudíž se ztratilo spojení se serverem. Mým úkolem bylo vytvořit skript, který bude restartovat PA na serveru, zjišťovat dobu provozu serveru od posledního restartu a vytvářet tiket hlášení.

4.2.2 Řešení

Jako první jsem musel zjistit, jak se restartuje PA pomocí NSH příkazů. Jelikož se cesta k PA liší, jak na Windowsu, tak i na UNIXu a UNIX má několik distribucí, tak jsem ještě musel zjišťovat cestu k PA na dané distribuci. Se zjišťováním cesty UNIXu mi pomohl kolega, který mi půjčil již hotovou funkci, kterou používají. Poté jsem musel zajistit kontrolu pro případ, kde PA není nainstalovaný, protože by jinak skript nefungoval. Když jsem zjistil cestu k PA a zkontroloval existující systém a distribucí, tak jsem spustil restart. Zjišťoval jsem, jestli PA funguje, pokud ne, tak jsem ho pouze nastartoval.

Windows: nexec -e cmd /c "net stop patrolagent"

nexec -e cmd /c "net start patrolagent"

U UNIX systému musel být příkaz spuštěn s uživatelem a identifikačním číslem procesu PA. Uživatele jsem hledal v souboru pomocí *`cat /etc/passwd`* a proces pomocí *`nexec -e "ps -ef | grep -e \"^.*PatrolAgent -p 3181 -id ${SERVER_NAME}\""`*.

UNIX: nexec –e „kill -9 ${id_process}“

nexec –e „/bin/su - ${user} -c \"cd /opt/bmc/Patrol3;./PatrolAgent -p 3181 –id ${SERVER_NAME}\"“

Po restartu jsem kontroloval pomocí funkce zjištění stavu, jestli už PA nastartoval nebo ještě startuje. Jakmile PA nastartoval, tak jsem zjišťoval dobu provozu serveru od posledního restartu

WINDOWS: patrolcli \"user ${username} ${password}\" \"open localhost 3181\" \"execpsl get(\\\"/NT_SYSTEM/NT_SYSTEM/SYSsysSystemUpTime/value\\\");\"\r

UNIX: export PATROL_HOME=\"/opt/bmc/Patrol3/$L_PATROL_HOME\"

PATH=${PATROL_HOME}/bin:${PATH}

/opt/bmc/Patrol3/$L_PATROL_HOME/bin/PatrolCli \"user ${username} ${password}\" \"open localhost 3181\" \"execpsl get(\\\"/UNIX_OS/UNIX_OS/SystemUptime/value\\\");\"

Po zajištění restartu a zjištění doby provozu od posledního restartu, se vytvořil tiket hlášení, který kontroluje, jestli monitoring na daný server pomocí PA funguje v pořádku a výsledky se vypsaly do tiketu.

WINDOWS: patrolcli \"user ${username} ${password}\" \"open localhost 3181\" \"execpsl set(\\\"/AS_EVENTSPRING/EVENTSPRING/AlertTest/value\\\",\\\"99\\\");\"\r

ping -n 15 127.0.0.1 > nul

patrolcli \"user ${username} ${password}\" \"open localhost 3181\" \"execpsl set(\\\"/AS_EVENTSPRING/EVENTSPRING/AlertTest/value\\\",\\\"0\\\");\"\r

UNIX: export PATROL_HOME=\"/opt/bmc/Patrol3/$L_PATROL_HOME\"

PATH=${PATROL_HOME}/bin:${PATH}

/opt/bmc/Patrol3/$L_PATROL_HOME/bin/PatrolCli \"user ${username} ${password}\" \"open localhost 3181\" \"execpsl set(\\\"/AS_EVENTSPRING/EVENTSPRING/AlertTest/value\\\",\\\"99\\\");\"

sleep 20

/opt/bmc/Patrol3/$L_PATROL_HOME/bin/PatrolCli \"user ${username} ${password}\" \"open localhost 3181\" \"execpsl set(\\\"/AS_EVENTSPRING/EVENTSPRING/AlertTest/value\\\",\\\"0\\\");\"

4.3 Kontrola zálohovacího softwaru

4.3.1 Zadání

Kontrola zálohovacího software byla jedna část ze skriptu velkého projektu AC425. Když se nastaví nový server, proběhne kontrola, která spouští tento skript. Skript kontroluje, jestli je vše správně nastaveno, jak na serveru, tak i v systému, aby se mohl server převzít do ostrého nasazení a mohl se používat. Kontroluje se zálohovací systém, antivirové softwary, hardwarová konfigurace a další. Mým úkolem bylo zajistit část skriptu, která měla kontrolovat funkčnost zálohovacího softwaru a samozřejmě, jestli tam takový software vůbec existuje.

4.3.2 Řešení

Jako první jsem musel zjistit, jak jsou takové zálohovací systému na serverech nainstalovány. Jakmile jsem měl všechny informace, začal jsem psát kód, ve kterém jsem kontroloval dvě věci a použil jsem pouze PowerShell. Zálohovací softwary byly čtyři typy (Hiback, LegatoNetWorker, NetBackup, TSM) a každý se kontroloval jinak.

U každého softwaru jsem kontroloval, jestli existuje na serveru.

Get-Service -Name "hibserv" -ErrorAction SilentlyContinue –ErrorVariable WindowsServiceExistsError

Pokud existoval, zjišťoval jsem ještě status softwaru, jestli běží nebo tam je chyba.

$ServiceStatus = (Get-Service -Name $ServiceName).Status

U zálohovacího softwaru Hiback jsem musel kontrolovat nejen status, ale taky hodnoty v souboru, do kterého zapisoval různé hodnoty, které udávaly, jestli funguje vše pořádku.

Kontrola zálohovacího softwaru LegatoNetWorker bylo nejrychlejší, protože nic navíc se kontrolovat nemuselo.

U zálohovacího softwaru NetBackupu bylo potřeba zjistit, jestli systém běží na virtuálním nebo na fyzické serveru.

$ComputerSystem = Get-WmiObject Win32_ComputerSystem | Select-Object Model, Manufacturer

Při kontrole zálohovacího softwaru TSM se ještě musely zjišťovat dvě hodnoty ze souboru, ve kterém bylo nastavení softwaru

4.4 Hiback 213

4.4.1 Zadání

Úkolem bylo vytvořit skript, který by spouštěl různé příkazy na čištění disku, spustil by zálohovací software Hiback a následně by zkontroloval aplikační logy ve Windows systému, jestli vše proběhlo v pořádku. Skript by se spouštěl pomocí ABX automaticky po kontrole po přijetí tiketu s problémem na zálohovací software Hiback s eror číslem 213.

4.4.2 Řešení

Po zjištění informací, jak má probíhat kontrola, když přijde takový tiket, jsem musel nastavit ABX s kolegou, který měl větší kompetence ve firmě než já. Nastavili jsme ho tak, že když přijde takový tiket s popisem, který obsahuje url odkaz, tak ho otevře. Po otevření odkazu zkontroluje, jestli obsah url odkazu obsahuje hodnoty, které určují, jestli je vše v pořádku nebo není. Pokud by bylo vše v pořádku, tak by se skript nespouštěl. Naopak pokud by hodnoty neodpovídaly, tak by se automaticky pomocí ABX spustil první skript. Tento skript kontroluje, jestli je operační systém serveru Windows a jestli je ve verzi 2008 a vyšší. Po téhle kontrole skript spouští pomocí NSH sadu příkazů, které maže nepotřebné soubory na disku a následně spustí zálohovací software pomocí příkazu

„nexec -e cmd /c 'C:\Hiback\Scripts\DoSystemState.cmd' &“

Průběh spouštění příkazů, jestli je vše v pořádku, se zapsal do poznámek k tiketu.

Toto zálohování potrvá nějakou chvíli a až doběhne, tak se potom pomocí Magic Button spustí druhý skript, který kontroluje operační systém a registr. Operační systém kvůli verzi Windowsu, která musí být 2008 nebo vyšší. V registru se kontroluje verzi PowerShellu, jestli je novější, protože starší verze nepodporuje příkazy pro práci s aplikačními logy v systému. Pomocí NSH se spouští PowerShellový skript, pomocí kterého zkontroluju aplikační logy. Pomocí následovného příkazu zjistím, jestli proběhlo všechno v pořádku nebo nastala chyba. Výsledek kontroly logu se následovně vypíše do poznámek k tiketu.

Get-WinEvent Microsoft-Windows-Backup -MaxEvents 1 | Where-Object { $_.TimeCreated -gt $ TimeUpdated-and $_.Id -eq 1 } | Select-Object ID | Format-Table -AutoSize –HideTableHeaders

4.5 Obnovení souborů

4.5.1 Zadání

Úkolem bylo pomocí zálohovacího software obnovit soubor nebo více souborů ze zvolené složky se zvoleným nastavením v určitém dni a času na zadaném serveru podle tiketu a uložit daný soubor nebo více souborů do složky, kterou zadal zákazník do tiketu.

4.5.2 Řešení

Ze začátku se zdál tenhle úkol lehký, ale zjistil jsem, budu muset nějakým způsobem získat informace z problém tiketu s požadavky na obnovení souborů. Po chvíli řešení jsem s kolegou, který měl více zkušeností s ABX, nastavil ABX, pomocí kterého se zkopíroval celý požadavek z tiketu do textového souboru a uložilo ho na server. Musel jsem zkopírovat celý obsah požadavku, který byl uložen v textovém souboru, a získat dané informace. Požadavek se skládal z času, server, seznamu souborů k obnovení, cestu k původní složce, cestu ke složce pro obnovení a nastavení složek, co všechno se obnoví. Všechny informace jsem musel ořezat a uložit zvlášť do proměnných, ze kterých jsem později tvořil příkaz na obnovení.

Po dokončení skriptu jsem přišel na problém a to ten, že ne na všech serverech je zálohovací software TSM, která obnovuje soubory a složky. Takže jsem musel ještě udělat kontrolu, abych zjistil, jestli tam zálohovací software TSM je. Kontrolu jsem prováděl zjištěním hodnot v registru na serveru. Pokud tam nebyla, tak se skript ukončil. Když v registru odpovídaly hodnoty, musel jsem zjistit, kde je aplikace uložena a tudíž jsem zjišťoval i její cestu. Cesta na Windows systémech i na UNIX systémech se lišila. Pomocí tohoto příkazu se spouštělo obnovení souborů.

```
nexec -e cmd /c "cd \"//${SERVER_NAME}/${TSMCLIENTPATHU/:/}" && dsmc
restore -pitd=${RestoreFromDate}${RestoreCmd}${FilePath}${RestoreLocationCmd}“
```

Využil jsem zde pouze NSH. Než jsem skript stihnul dokončit na 100%, tak přišlo rozhodnutí, které nebylo příjemné a to takové, že už to nemusím skriptovat a není to potřeba.

5 Teoretické a praktické dovednosti a znalosti

5.1 Získané dovednosti a znalosti v průběhu studia uplatněné v průběhu odborné praxe

Největší dovednost a znalost, kterou jsem si osvojil v rámci studia a posléze uplatnil v Tieto Czech s.r.o., byly znalosti týkající se psaní kódu a logiky programování. Tyto znalosti a dovednosti jsem se naučil z programovacích předmětů:

- základy programování
- algoritmy 1, 2
- programovací jazyky 1, 2
- úvod do softwarového inženýrství
- úvod do databázových systémů
- databázové a informační systémy
- vývoj informačních systémů
- skriptovací programovací jazyky a jejich aplikace
- správa windows serverů

5.2 Scházející dovednosti a znalosti v průběhu odborné praxe

Při nástupu na odbornou praxi byla mým největším problémem neznalost programovacího jazyka NSH, který měl dokumentace a texty v anglickém jazyce, jenž nebyl zrovna mou nejlepší stránkou. Později bylo potřeba umět i PowerShell, ze kterého jsem uměl jenom základní znalosti, které jsem musel prohloubit. Dále nebylo lehké pochopit celý systém automatizace, abych mohl bez problémů a samostatně řešit problémy spojené s laděním nakódovaných skriptů.

6 Závěr a celkové výsledky

Možnost absolvování bakalářské práce v podobě odborné praxe ve firmě Tieto Czech s.r.o. mi umožnila jako studentovi vysoké školy s IT zaměřením rozšířit a zlepšit své znalosti a jejich využití v praxi. Naučil jsem se zde práci v kolektivu zkušených vývojářů, pracovat s novými postupy, moderní technologií a vyzkoušet si práci v agilním vývoji[1]. Také jsem získal reálnou představu o tom, jaké jsou možnosti uplatnění a s čím vším se lze v praxi setkat. Tato zkušenost mi velmi pomohla při rozhodování, kterým směrem se budu dále ubírat.

Po získání těchto zkušeností byla pro mě přínosnější odborná praxe než vykonávání bakalářské práce klasickou formou. Při klasické bakalářské práci bych nezískal takové zkušenosti jako na praxi, protože bych řešil sám pouze jeden problém, u kterého bych neměl jistotu, že se s takovým problémem setkám v praxi.

Doufám, že tato praxe byla přínosem jak pro mou profesní budoucnost, tak i přínosem pro společnost Tieto Czech s.r.o., a že jsem splnil očekávání, vyplývající ze splněných úkolů.

[1] agilní vývoj – agilní vývoj je interaktivní způsob řízení projektů. V praxi to znamená těsnou a neustálou (inkrementální) spolupráci mezi projektovým týmem, který vytváří průběžné prototypy a mezi zákazníkem, který dává zpětnou vazbu na základě které se upřesňuje zadání. Uplatňuje se v projektech, u kterých není jasný rámcový cíl, ale z nejrůznějších důvodů nelze přesně definovat všechny dlouhodobé požadavky bez průběžných prototypů.

7 Reference

[1] *BladeLogic Server Automation 8.5: Foundtation – Part 2, Lab Guide.* Houston, Texas: BMC Software, Inc., 2015.

[2] *BladeLogic Server Automation 8.5: Foundtation – Part 3, Lab Guide.* Houston, Texas: BMC Software, Inc., 2015

[3] *BladeLogic Server Automation 8.5: Foundtation – Part 2, Student Guide.* Houston, Texas: BMC Software, Inc., 2015

[4] *BladeLogic Server Automation 8.5: Foundtation – Part 3, Student Guide.* Houston, Texas: BMC Software, Inc., 2015

[5] *BladeLogic Server Automation 8.2: Foundtation – Part 2, Student Guide.* Houston, Texas: BMC Software, Inc., 2012

[6] *Tieto Czech s.r.o.: Webové stránky firmy* [online]. 2016 [cit. 2016-03-17]. Dostupné z: http://www.tieto.cz

[7] *SS64: Command line reference - Web, Databases and OS scripting* [online]. 2016 [cit. 2016-03-17]. Dostupné z: http://www.ss64.com

[8] *BMC: Bring IT To Life with Digital Enterprise Management* [online]. 2016 [cit. 2016-03-17]. Dostupné z: http://www.bmc.com

[9] *Microsoft TechNet: Resources and Tools for IT Professionals* [online]. 2016 [cit. 2016-03-17]. Dostupné z: https://technet.microsoft.com

[10] *Software Development and Collaboration Tools | Atlassian* [online]. 2016 [cit. 2016-03-17]. Dostupné z: https://www.atlassian.com